Data Mining

ID2222

Homework 3: Mining Data Streams

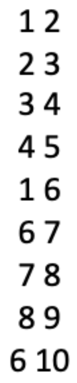
Authors:
Abdul Aziz Alkathiri
Francesco Staccone

November 26, 2019

1 Introduction

The dataset used to complete the task is one of the publicly available graph datasets taken from the link http://konect.uni-koblenz.de/networks/, in particular the "Air traffic control" one.

The dataset contains a long list of egdes (u, v), where u and v are two vertices of the graph built by all the edges in the dataset, as shown in Figure 2.



```
1 2
2 3
3 4
4 5
1 6
6 7
7 8
8 9
6 10
```

Figure 1: The edges dataset.

Our submission consists of one .ipynb file that includes the Reservoir Sampling algorithm studied during the lectures and then implements one of the algorithms in the proposed papers that leverage on the Reservoir Sampling to achieve their goal.

We chose to implement the solution proposed by "TRIÈST: Counting Local and Global Triangles in Fully-Dynamic Streams with Fixed Memory Size" (De Stefani L., Epasto A., Riondato M, Upfal E., 2017), a suite of one-pass streaming algorithms to approximate, at each time instant, the global and local number of triangles in a fully-dynamic graph stream (i.e., a sequence of edges additions and deletions in arbitrary order) using a fixed amount of memory.

In particular, we implemented the base and the improved versions.

2 Implementation

2.1 Reservoir Sampling

The Reservoir Sampling is a randomized sampling of specified size. We have M as the specified memory size chosen by the user and t as the specific element in the stream of edges at time t. While streaming edges, do continuously:

1. Keep first M items in memory;
2. With Probability (M/t) we keep the t-th item and discard one of the previous ones with the same probability;
3. Calls the triangles counters update with sign "-" in case of base version.

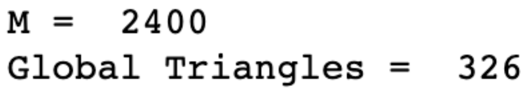
2.2 TRIÈST Algorithm

TRIÈST algorithm approximates the global and local number of triangles in a streamed graph. While streaming edges, do continuously:

1. If the received edge is not currently in the edge sample, do Reservoir Sampling and process the edge, otherwise skip the edge and do not process it.
2. For every processed edge call a counters' update:
   (a) check the common neighbours of the vertices of the current edge (intersection of their neighbours sets);
   (b) compute the update weight using this formula for the improved version: max {1, (t-1)(t-2)/M(M-1)}; otherwise add or remove 1 based on the sign (+ or -) received in the base version;
   (c) update local counters for each vertex and also the global triangle counter.
3. Return the global number of triangles variable value for the improved version; otherwise, return the global number of triangles variable value multiplied by the value: max {1, t(t-1)(t-2)/M(M-1)(M-2)} for the base version.

2.3 Results

The proposed results are the ones produced by the improved version but it's sufficient to change the boolean "improved" from True to False to compute the results for the base version. In particular, with the improved version we found out that if we select M ¿= 2400, the number of estimated triangles in the graph is 326, as visible in Figure 2.



```
M =  2400
Global Triangles =  326
```

Figure 2: Results of the improved version based on M value.

3 Questions

1. What were the challenges you have faced when implementing the algorithm?

   The biggest challenge was to implement the base algorithm, since it required some time to understand how to manage the neighbours and to implement the whole available pseudocode, then the improved TRIÈST algorithm was faster to implement even though there was no pseudocode for it (just a few lines changed with respect to the base verision).

2. Can the algorithm be easily parallelized? If yes, how? If not, why? Explain.

   Yes, it is possible to parallelize the edge processing and counters update easily, using global counters and shared edge sample. Both Reservoir Sampling and TRIÈST with counters update can be computed in parallel, using shared memory.

3. Does the algorithm work for unbounded graph streams? Explain.

   Yes, since the idea of Reservoir Sampling is to mine potentially infinite streams of data, without taking into account the stream length. TRIÈST dynamically updates the counters of global and local triangles and processes each edge independently and in relation with only the ones present in the memory.

4. Does the algorithm support edge deletions? If not, what modification would it need?Explain.

The version we implemented does not support edge deletions but only insertions. To support deletions, a modification of TRIÈST-base called TRIÈST-FD is needed, that is based on random pairing (RP), a sampling scheme that extends Reservoir Sampling and can handle deletions. In particular, it needs a tracker that monitors the uncompensated edge deletions and decreases the number of triangles if necessary. It also needs modification on Reservoir Sampling to support edge deletions in the dataset, by tracking if the edge-to-be-deleted is in the current edge list or not, computing a different probability of insertion. If the edge-to-be-deleted was in the sampled list, the algorithm will compensate the deletion with an immediate insertion.